ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.12, December 2009, pp.3193−3204
doi: 10.3724/SP.J.1001.2009.03400
© by *Institute of Software, the Chinese Academy of Sciences*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

网络受限移动对象过去、现在及将来位置的索引*

丁治明+, 李肖南, 余 波

(中国科学院 软件研究所 基础软件国家工程研究中心,北京 100190)

Indexing the Historical, Current, and Future Locations of Network-Constrained Moving Objects

DING Zhi-Ming+, LI Xiao-Nan, YU Bo

(Fundamental Software Research Center, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: zhiming@iscas.ac.cn

Ding ZM, Li XN, Yu B. Indexing the historical, current, and future locations of network-constrained moving objects. *Journal of Software*, 2009,20(12):3193−3204. http://www.jos.org.cn/1000-9825/3400.htm

Abstract: A new index structure for network constrained moving objects, network-constrained moving objects dynamic trajectory R-Tree (or NDTR-Tree for short), is proposed in this paper. NDTR-Tree can index the whole trajectories of moving objects, including their historical, current, and near future positions. In order to compare the performance of the NDTR-Tree with other index structures for network constrained moving objects, such as MON-Tree and FNR-Tree, a series of experiments have been conducted, and the experimental results show that the NDTR-Tree outperforms the previously proposed network constrained moving objects index methods in dealing with dynamically maintained trajectories of moving objects.

Key words: moving object; database; index; spatial-temporal trajectory

摘 要: 提出了一种适合于网络受限移动对象数据库的动态轨迹 R 树索引结构(network-constrained moving objects dynamic trajectory R-Tree,简称 NDTR-Tree).NDTR-Tree 不仅能够索引移动对象的整个历史轨迹,而且能够动态地索引和维护移动对象的当前及将来位置.为了比较相关索引结构及算法的性能,进行了详细的实验.实验结果表明,与现有的基于道路网络的移动对象索引方法如 MON-Tree 和 FNR-Tree 等相比,NDTR-Tree 有效地提高了对网络受限移动对象动态全轨迹的查询处理性能.

关键词: 移动对象;数据库;索引;时空轨迹

中图法分类号: TP311 **文献标识码**: A

近年来,作为移动计算技术的重要分支以及位置相关类应用的底层支撑技术之一,移动对象数据库(moving objects databases,简称MOD)技术得到了广泛的重视与研究.移动对象数据库是指对移动对象的位置及其他相关信息进行表示与管理的数据库[1],在移动对象数据库中通常管理着大量的移动对象,这些移动对象的位置是不断变化的,如汽车、飞机、移动用户等.越来越多的应用要求对移动对象进行管理,而定位技术和无线通信技

* Supported by the National Natural Science Foundation of China under Grant No.60573164 (国家自然科学基金); the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry (国家教育部留学回国人员科研启动基金)

Received 2007-08-27; Revised 2008-02-01; Accepted 2008-05-19

© 中国科学院软件研究所 http://www.c-s-a.org.cn

术的发展使得跟踪和记录移动对象的位置成为可能.

在典型的移动对象数据库系统中,通常存放着海量的移动对象时空轨迹数据.例如,一个大中型城市的移动对象数目可以达到数十万甚至更多.为了支持对这些移动对象过去及当前位置的查询,有效的索引手段是需要首先解决的关键问题.

在移动对象索引方面,人们已经进行了许多工作.文献[2]在基本R*树索引的基础上,提出了一种TPR-Tree移动对象索引结构.TPR-Tree中的最小包容矩形(minimum bounding rectangle,简称MBR)含有速度等参数信息,能够随着时间参数进行变化.TPR-Tree提出后,人们在此基础上进行了大量的改进工作[3−6],其中代表性的成果是文献[3]中提出的TPR*-Tree.实验结果表明,TPR*-Tree的性能与TPR-Tree相比得到了大幅度的提高.但是,由于没有存放历史轨迹信息,TPR-Tree及其变种只能支持移动对象当前以及短暂将来位置的查询,不能支持对过去位置的查询.文献[7]提出了一种Map-Based R-Tree(MBR-Tree)索引,通过将移动对象和静态的地图空间对象相关联来组织索引结构,从而有效地支持基于地物的查询,但该方法只能处理移动对象的当前位置.为了支持移动对象过去位置的查询,需要对移动对象完整的时空轨迹进行索引.在这方面,代表性的成果包括文献[8−10]等.然而由于历史轨迹属于静态信息,上述方法没有考虑运行中的移动对象数据库的关键问题,即活动轨迹单元的动态维护问题,因此仅能支持对移动对象过去位置的查询,无法支持对当前及将来位置的查询.文献[11]研究了移动对象未来轨迹的动态维护问题,但移动对象需要预先提交其运行计划.所有上述工作均直接基于Euclidean空间,由于没有考虑道路网络和移动对象的交互作用关系,因此影响了其查询性能的进一步提高[12].

近年来,人们逐渐认识到网络受限移动对象的重要性,并提出了若干基于交通网络的移动对象数据库索引.文献[12]提出了一种 FNR-Tree(fixed network R-tree)索引结构.FNR-Tree 分为上下两层,其中上层是一个 2 维的 R 树,用于对固定的道路网络进行索引;下层是一系列的 1 维 R 树,用于对道路网络中的移动对象进行索引.文献[12]的重要贡献之一在于通过实验证明了与直接基于 Euclidean 空间的移动对象索引方法相比,基于路网的移动对象索引方法通过上下分层而显著地提高了查询处理的效率.然而,FNR-Tree 在对道路网络进行表示和索引时是以单个直线线段为基本单位的,从而造成了巨大的记录数目.文献[13]提出了一种基于 Hilbert 曲线的网络受限移动对象索引方法,该索引结构与 FNR-Tree 相似,也是直接基于直线线段进行处理的,因此有着相同的缺陷.为了克服 FNR-Tree 的缺陷,文献[14]提出了一种 MON-Tree(moving objects in networks tree)索引结构.MON-Tree 对 FNR-Tree 的最大改进在于,对交通网络的组织不再以直线线段为单位,而是以折线(polyline)表示的道路(route)或者边(edge)为单位.这样不仅减少了记录的个数,而且降低了在表示移动对象跨越不同下层 R 树的工作量.实验结果表明,MON-Tree 比 FNR-Tree 具有更好的性能.然而 MON-Tree 仍然存在着一些缺陷,主要表现在上下两层只能选择相同的网络模型(edge-based 或 route-based).当选择 edge-based 模型时,下层 R 树的数目较多,且需要使用较多的数据来表示移动对象从一条边向另一条边的转换;反之,当选择 route-based 模型时,上层 R 树的 MBR 会有大量的重合,将直接导致查询性能的下降.此外,所有上述网络受限移动对象索引均只能处理历史轨迹,不能支持对移动对象当前位置的查询.文献[15,16]对网络受限移动对象的当前和将来位置的索引进行了研究.但是,上述方法不能用于索引移动对象的历史轨迹.

通过上述分析可以看出,在网络受限移动对象的索引技术方面,目前缺乏一种能够对移动对象的过去、现在以及将来位置进行统一表示和查询的处理方法.为了解决上述问题,本文提出了一种适合于网络受限移动对象数据库的动态轨迹 R 树(network-constrained moving objects dynamic trajectory R-tree,简称 NDTR-Tree)索引结构.

NDTR-Tree 的主要目标是为了对网络受限移动对象的过去、现在以及将来位置提供索引支持.尽管文献[6,17]已经对移动对象过去、现在及将来位置的索引问题进行了研究,但它们均是直接基于 Euclidean 空间的,没有针对网络受限的环境进行必要的优化.NDTR-Tree 采用了目前基于路网的移动对象索引中所通用分层结构,将静态的路网与动态移动对象进行分离处理.此外,NDTR-Tree 通过与移动对象的位置更新操作相结合,实现了活动运行矢量(其中包含移动对象的当前及将来位置信息)在索引中的显式表示及动态维护,从而为移动对象的查询提供了全方位的支持.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

NDTR-Tree 的另一个目标是提高移动对象索引结构在引入了活动运行矢量动态维护机制之后的效率.目前的网络受限移动对象索引方法(如 FNR-Tree,MON-Tree 等)均是基于静态的历史时空轨迹的,在表示道路网络时采用的是单一粒度的方法,不能有效地满足索引动态维护当前位置信息的要求.为了解决上述问题,NDTR-Tree 采用了一种 Hybrid 结构,上层 R 树以粒度较小的原子路段为基本记录单位,极大地减少了 MBR 的重合度;而下层 R 树以粒度较大的道路来进行组织,降低了移动对象跨越不同下层 R 树时索引维护的工作量,从而有效地提高了索引维护及查询处理的效率.

本文第 1 节给出网络受限移动对象的时空轨迹模型.第 2 节描述 NDTR-Tree 索引的结构及相关算法.第 3 节对实验结果进行分析.第 4 节给出相关结论.

1 网络受限移动对象的时空轨迹模型

我们首先给出基于原子路段、道路以及交叉路口的交通网络数据模型,然后定义网络受限移动对象的时空轨迹,最后对时空轨迹的生成方法以及轨迹曲线进行分析.

在对交通网络进行表示时,我们采用了一种道路-原子路段的双层结构.与以往提出的单层路网模型[14,18]相比,该模型能够在兼顾移动对象位置更新效率的前提下,提高索引粒度选择的灵活性.

定义 1(交通网络). 交通网络 G 定义为

$$G=(R,J),$$

其中,R 是道路的集合,J 是交叉路口的集合.

定义 2(道路). 道路 r 定义为如下形式:

$$r=(rid,geo,len,(jid_j,pos_j)_{j=1}^{m},ARS),$$

其中,rid 是道路标识;geo 是该道路的地理几何形状,len 是道路的长度;$(jid_j,pos_j)_{j=1}^{m}$ 是该条道路所包含的各个交叉路口(见定义 3)的标识以及它们在道路中的相对位置(设每条道路的总长度为 1,则该条道路中的任一相对位置 pos 可以用[0,1]之间的一个实数表示),ARS 是该条道路原子路段(见定义 4)的集合.

在上述定义中,表示道路几何形状 geo 的折线 pl 是由 $X\times Y$ 平面中的一组点所组成的序列,即

$$pl=(x_i,y_i)_{i=1}^{n}\ (n\geq 2),$$

其中,(x_1,y_1)为pl的起点,或称 0-端点,(x_n,y_n)为pl的终点,或称 1-端点.

定义 3(交叉路口). 交叉路口 j 与实际交通网络中的交叉路口、出入口或道路端点相对应,j 定义为如下形式:

$$j=(jid,loc,((rid_i,pos_i))_{i=1}^{n},m),$$

其中,jid是交叉路口的标识;loc是j的地理位置,用其(x,y)坐标表示;$(rid_i,pos_i)(1\leq i\leq n)$描述$j$所连接的第$i$条道路,其中,$rid_i$是道路标识,$pos_i\in[0,1]$是$j$在该条道路中的相对位置;$m$是$j$的连接矩阵[18],用以描述该交叉路口的各交通流之间的连通关系.

定义 4(原子路段). 原子路段(atomic route section)是对道路交通流进行分割的基本单位.对于道路 r,如果它是双向道路,则有两个交通流:$r+$和 $r-$;如果是单行道路,则只有一个交通流,$r+$或 $r-$.原子路段 ars 则是 $r+$或 $r-$中连接两个相邻交叉路口或道路端点的基本单位,且其中不再含有任何允许移动对象离开该交通流的其他交叉路口.ars 定义为如下的三元组:

$$ars=(aid,(jid_s,pos_s),(jid_e,pos_e)),$$

其中,aid是该原子路段的标识;(jid_s,pos_s)和(jid_e,pos_e)分别是原子路段的起点和终点所对应的交叉路口及其在道路中的相对位置.原子路段所属的交通流方向通过jid_s和jid_e指定,即允许移动对象从pos_s向pos_e方向行驶.

图 1 给出了道路及原子路段的例子.图中的道路r_1是双向道路,因此具有两个交通流r_1+(从 0-端点驶向 1-端点)和r_1-(从 1-端点驶向 0-端点).原子路段以交叉路口为结点对r_1+和r_1-进行分割,因此r_1包含 8 个原子路段.由于r_1中的交叉路口全部是r_1+和r_1-公用的,因此上述 8 个原子路段是两两对称的.如果道路中存在只属于某个方向交通流的交叉路口(如图 2 所示),则道路双边的原子路段可以不对称.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

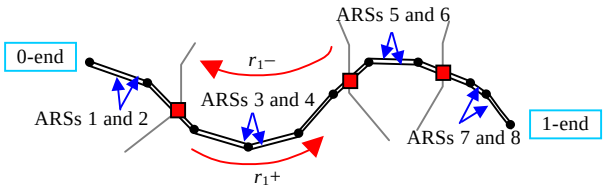


Fig.1 Route r_1 and its atomic route sections

图 1 道路r_1及其原子路段

上述定义给出了一种道路网络的基本描述方法.通过道路、交叉路口、原子路段等数据类型,系统可以描述复杂的道路网络及其交通流连接关系.图 2 给出了一个实际道路网络描述的例子(道路r_1中的每个箭头代表一个原子路段,r_1双向共含有 10 个原子路段).

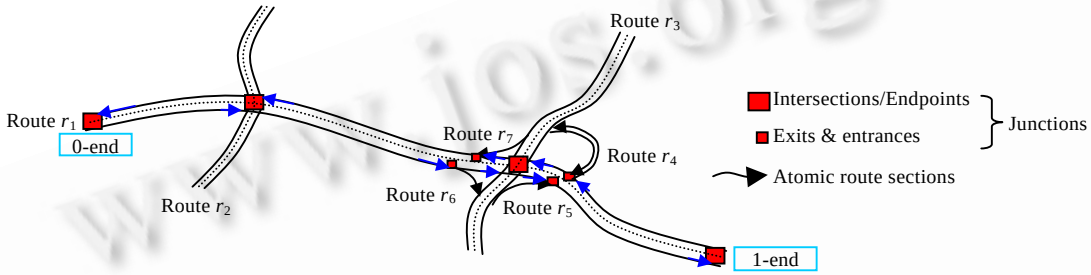


Fig.2 A complicated transportation network representation

图 2 复杂交通网络的表示

定义 5(网络位置). 交通网络 G 中的任意一个网络位置(graph position)*gpos* 可用其所在的道路标识 *rid* 及其在该道路中的相对位置 *pos* 表示,即

$$gpos=(rid,pos),$$

其中,*rid* 是道路标识,$pos\in[0,1]$是 *gpos* 在该道路中的相对位置.由于每条道路的几何形状都以折线的方式存放在数据库中(见定义 2),上述表示方法可以很容易地转换为(x,y)的形式.

定义 6(移动对象的运行矢量). 移动对象 *mo* 在 *t* 时刻的运行矢量(motion vector)*mv* 定义为如下形式:

$$mv=(t,(rid,pos),\vec{v}),$$

其中,*t* 是采集该运行矢量的时间;(*rid*,*pos*)是移动对象 *mo* 在 *t* 时刻的网络位置;$\vec{v}$ 是一个带符号的速度,其符号表示 *mo* 的行驶方向(由 0-端向 1-端行驶符号为+,反之为–),其值为 *mo* 在 *t* 时刻的速度.

当移动对象在交通网络中行驶时,需要不断地将当前运行参数(如位置、速度、方向等)与上一次位置更新时所提交的运行矢量$mv_n=(t_n,(rid_n,pos_n),\vec{v}_n)$进行比较.一旦某些预先定义的位置更新条件满足时,就需要触发一次位置更新过程,将其最新的运行参数发送给服务器.

在网络受限的移动对象数据库中,文献[18]定义了 3 种类型的位置更新,即道路ID触发的位置更新(IDTLU)、距离阈值触发的位置更新(DTTLU)以及速度阈值触发的位置更新(STTLU).其中,DTTLU和STTLU分别在移动对象超出距离阈值ξ和速度阈值ψ时进行触发,并分别产生一个运行矢量mv_a;IDTLU在移动对象从一条道路r_s转换到另一条道路r_e时进行触发,并将产生 3 个运行矢量 $mv_{a_1},mv_{a_2},mv_{a_3}$.其中, mv_{a_1} 和 mv_{a_2} 分别对应于移动对象在通过交叉路口时的状态(分别以交叉路口在r_s和r_e中的位置表示), mv_{a_3} 对应于移动对象在触发 IDTLU 时的状态.上述 3 种位置更新同时作用,共同完成移动对象时空轨迹(见定义 7)数据的采集过程.

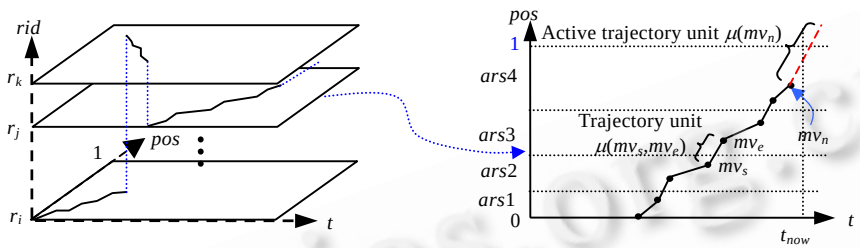
定义 7(移动对象的时空轨迹). 移动对象 *mo* 的时空轨迹 *trajectory* 是 *mo* 在行驶过程中通过位置更新操作所提交的一组运行矢量的序列,定义为如下形式:

$$trajectory=(mv_i)_{i=1}^{n}=((t_i,(rid_i,pos_i),\vec{v}_i))_{i=1}^{n},$$

© 中国科学院软件研究所 http://www.c-s-a.org.cn

其中,mv_n是移动对象所提交的最后一个运行矢量,称为活动运行矢量.活动运行矢量包含计算移动对象现在及将来位置的关键信息.

通过移动对象的时空轨迹,我们可以计算出移动对象在任一指定时刻t_q的位置.如果t_q处于两个连续的运行矢量之间,则可采用插值的方法进行计算;若t_q在活动运行矢量之后,则可以通过活动运行矢量中的相关信息进行推算[18].从这个意义上讲,移动对象时空轨迹实际上刻画的是$RID\times POS\times T$空间中的一条不间断的曲线,如图 3 所示.



(a) Trajectory of network-constrained moving object (b) Trajectory segment inside route r_j

(a) 网络受限移动对象的轨迹 (b) 在道路r_j中的轨迹片段

Fig.3 Spatial-Temporal trajectories of network-constrained moving objects

图 3 网络受限移动对象的时空轨迹

我们称上述时空轨迹曲线中的每一条线段为一个轨迹单元.两个相邻的运行矢量mv_s和mv_e所对应的轨迹单元记为$\mu(mv_s,mv_e)$,它是连接mv_s和mv_e对应点的一条线段;活动运行矢量$mv_a=(t_a,(rid_a,pos_a),\vec{v}_a)$对应的轨迹单元记为$\mu(mv_a)$,它是以$mv_a$所对应的点为端点的一条射线(在根据上述射线计算移动对象在当前时刻t_{now}的位置时,超出 1 的部分以 1 作为返回值[1,18]).

2 NDTR-Tree 索引的结构及相关算法

本节首先给出 NDTR-Tree 的结构,然后分析 NDTR-Tree 中活动轨迹单元的处理方法,最后给出相关的索引维护与查询处理算法.

2.1 NDTR-Tree索引的结构

NDTR-Tree 的结构分为上下两层.上层为一个 R 树,用于对交通网络的道路数据以原子路段为单位进行索引;下层为一组独立的 R 树,每个 R 树与一条道路相对应,用于对各移动对象在该条道路上提交的时空轨迹数据以轨迹单元为单位进行索引.与其他基于交通网络的移动对象索引结构(如 FNR-Tree,MON-Tree 等)相比,NDTR-Tree 是一种 Hybrid 结构,其上层 R 树的基本索引单位是原子路段(粒度等同于文献[14]中的 edge-based 模型),而每个下层 R 树对应的是一条道路(粒度等同于文献[14]中的 route-based 模型).因此,上层 R 树叶子结点记录与下层 R 树之间的对应关系是 n:1 的关系,而不是 FNR-Tree 和 MON-Tree 中的 1:1 关系.

选择原子路段而不是道路作为上层 R 树索引的基本记录单位的原因是为了减少上层 R 树中各最小包容矩形(minimum bounding rectangle,简称 MBR)的重合度.在交通网络中,有的道路如北京的长安街、阜石路等可以横跨大半个城市,在 R 树中将造成大量 MBR 的重合,从而极大地降低了查询处理的效率.因此,选择较小的空间对象(如原子路段)作为索引记录的基本单位,将有利于提高上层 R 树的查询效率.

另一方面,根据道路而不是原子路段来组织下层R树的原因在于,选择较大的空间单位来组织有利于减少下层R数的个数,并降低移动对象在跨越不同的下层R树时的维护代价[14].此外,选择道路为单位能够与网络受限移动对象的位置表示方法及位置更新算法[18]兼容.图 4 给出了NDTR-Tree的结构.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

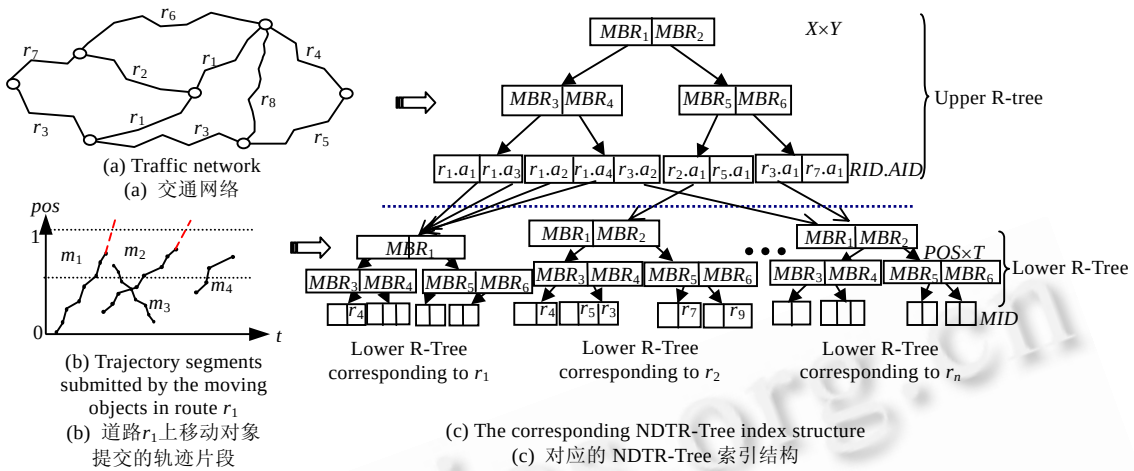


Fig.4 Structure of the NDTR-Tree

图 4 NDTR-Tree 的结构

如图 4 所示,NDTR-Tree的上层R树是一个以原子路段为基本索引单位的标准R树结构,其中间结点中的记录项为$\langle MBR_{xy},PT_{node}\rangle$,$MBR_{xy}$是$X\times Y$平面中的MBR,$PT_{node}$是指向下层结点的指针;其叶子结点中的记录项为$\langle MBR_{xy},rid.aid,PT_{route},PT_{tree}\rangle$,其中,$MBR_{xy}$是原子路段的MBR,$rid.aid$是原子路段及其所属道路的标识, PT_{route}是指向道路详细记录的指针,PT_{tree}是指向下层R树的指针.

下面让我们来考察NDTR-Tree的下层R树结构.如前所述,每棵下层R树对应于一条具体的道路r(设其标识为rid),用于索引在r上行驶的移动对象在位置更新时所生成的轨迹单元.下层R树的中间结点中的记录项为$\langle MBR_{pt},PT_{node}\rangle$,其中,$MBR_{pt}$为$POS\times T$平面中的MBR,$PT_{node}$是指向下层结点的指针;其叶子结点中记录项的结构为$\langle mid,mv_s,mv_e,MBR_{pt}\rangle$,其中,$MBR_{pt}$为轨迹单元在$POS\times T$平面中的MBR,$mid$为移动对象的标识,$mv_s=(t_s,rid,pos_s,\vec{v}_s)$和$mv_e=(t_e,rid,pos_e,\vec{v}_e)$分别为不确定轨迹单元对应的两个连续的运行矢量.对于活动轨迹单元而言,mv_e为空且MBR_{pt}为$\langle t_n,pos_n,t_s,pos_s\rangle$.

2.2 NDTR-Tree索引中活动轨迹单元的处理

在NDTR-Tree中,为了支持对移动对象当前位置的查询,需要将其活动轨迹单元存放在索引结构中.由于活动轨迹单元$\mu(mv_n)$实际上是一条以(pos_n,t_n)为端点的射线l(l的斜率取决于$\vec{v}_n$),因此在索引中需要保存该射线多大的长度是需要首先确定的问题.在下面的讨论中,设mo从道路0-端向1-端行驶.mo从1-端向0-端行驶的情况可以通过类似的方法进行分析.

由分析可知,l与直线pos=1的交点为$(t^*,1)$,其中,t^*可如下式求得($r.length$为移动对象所在道路r的长度):

$$t^* = t_n + \frac{(1-pos_n)\times r.length}{|\vec{v}_n|}.$$

如果移动对象mo在到达道路端点之前没有发生位置更新(即按照预测的轨迹行进),则可有如下两种情况:

(1) mo的实际平均速度$\vec{v}\geq\vec{v}_n$,则在到达t^*时刻之前,mo就会触发IDTLU(设触发IDTLU的时间为t_u),因此,在计算移动对象的当前位置时,只需要使用到射线l位于直线pos=1之下的部分即可(见图5(a)中l的实线部分);

(2) mo的实际平均速度$\vec{v}<\vec{v}_n$,根据网络受限移动对象的位置更新协议[18]可知,mo的最慢平均速度为$\vec{v}_s=\vec{v}_n-\psi$(其中,ψ为速度阈值).以$\vec{v}_s$到达道路1-端点的时刻t_s为

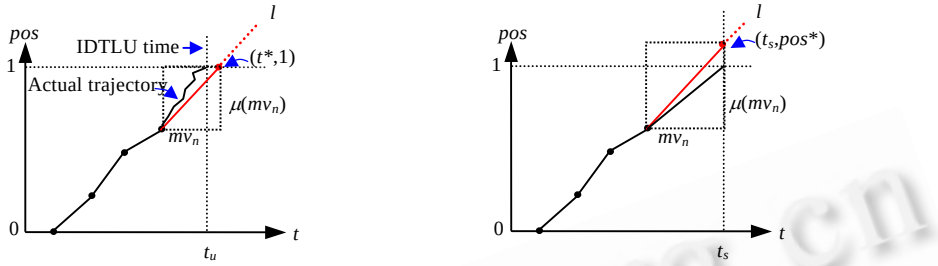
$$t_s = t_n + \frac{(1-pos_n)\times r.length}{|\vec{v}_s|}.$$

l与直线$t=t_s$的交点为(t_s,pos_s),其中,pos_s可如下式求得:

© 中国科学院软件研究所 http://www.c-s-a.org.cn

$$pos_s = pos_n + \frac{(t_s - t_n)\times|\vec{v}_n|}{r.length}.$$

在计算移动对象的当前位置时,只需要l位于点(t_s,pos_s)之下的部分即可(如图 5(b)所示l的实线部分).



(a) *mo*'s actual speed>v_n (b) *mo*'s actual speed<v_n

(a) *mo*运行速度>v_n (b) *mo*运行速度<v_n

Fig.5 Determining the MBRs of active trajectory units

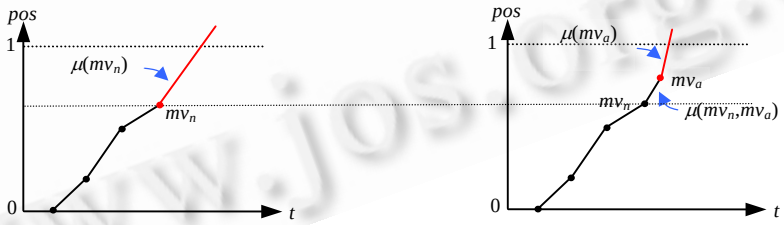
图 5 活动轨迹片段 MBR 的确定

综合上述情况(1)、情况(2)的分析,在NDTR-Tree索引中,仅需保留活动轨迹单元对应的射线l中位于$MBR=\langle t_n,pos_n,t_s,pos_s\rangle$中的线段,即可满足对移动对象现在及将来位置查询的要求.其中:

$$t_s = t_n + \frac{(1-pos_n)\times r.length}{|\vec{v}_s|},\ pos_s = pos_n + \frac{(t_s - t_n)\times|\vec{v}_n|}{r.length}.$$

下面考察发生位置更新时的情况.由于活动轨迹单元中含有预测信息,因此,当移动对象偏离预测轨迹并发生位置更新时,就需要对相应的索引记录进行调整.

首先分析发生DTTLU和STTLU的情况.由网络受限移动对象的位置更新协议可知,在发生DTTLU和STTLU时,移动对象仍然在原先的道路上行进.当服务器接收到一个新的位置更新消息$mv_a=(t_a,rid_a,pos_a,\vec{v}_a)$时,如果$mv_a$是$rid_a$所对应的下层R树中该移动对象的第 1 个运行矢量,则直接将$\mu(mv_a)$插入该下层R树即可;否则,需要对下层R树进行如下处理:(1) 删除上一次位置更新时产生的活动轨迹单元$\mu(mv_n)$;(2) 插入$\mu(mv_n,mv_a)$;(3) 插入$\mu(mv_a)$,如图 6 所示.



(a) Trajectory in NDTR-Tree at t_n (b) Trajectory in NDTR-Tree at t_a

(a) 在t_n时刻NDTR-Tree中的轨迹 (b) 在t_a时刻NDTR-Tree中的新轨迹

Fig.6 Maintenance of the bottom R-Trees in NDTR-Tree when DTTLU or STTLU occurs

图 6 NDTR-Tree 在发生 DTTLU 和 STTLU 时对下层 R 树的维护

当发生IDTLU时,处理过程要相对复杂一些.设移动对象从道路r_s转换到r_e,此时将产生 3 条位置更新消息$mv_{a_1},mv_{a_2},mv_{a_3}$.其中,$mv_{a_1}$对应于$r_s$,$mv_{a_2}$和$mv_{a_3}$对应于$r_e$,且$mv_{a_1}$和$mv_{a_2}$对应于移动对象在交叉路口时的状态.NDTR-Tree需要进行的操作是:(1) 在r_s对应的下层R树中删除$\mu(mv_n)$,并插入$\mu(mv_n,mv_{a_1})$;(2) 在r_e对应的下层R树中插入$\mu(mv_{a_2},mv_{a_3})$;(3) 在r_e对应的下层R树中插入$\mu(mv_{a_3})$.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

2.3 NDTR-Tree索引的建立、维护与查询算法

移动对象数据库在建立NDTR-Tree时,需要首先扫描交通网络数据中的道路记录并建立上层R树索引.当上层R树初始建立时,各叶子结点记录中的PT_{tree}指针所指向的下层R树均为空树.

当移动对象向服务器发送位置更新消息时,系统需要将相应的轨迹单元插入到相应的下层 R 树中.如前所述,当发生 DTTLU 和 STTLU 时,需要在对应的下层 R 树中删除原活动轨迹单元,并插入两条新的记录;当发生 IDTLU 时,需要在两个不同的下层 R 树中进行相应的删除和插入操作.

NDTR-Tree 的建立及维护算法见算法 1.函数 *mbr*(·)返回移动对象轨迹单元的 MBR, *Insert*(·)和 *Delete*(·)分别在下层 R 树中执行记录的插入和删除操作.

算法 1. NDTR-Tree 的建立及维护算法.

变量说明:$G=(R,J)$; //交通网络

1. 扫描交通网络中的道路集合 R,以原子路段为记录单位建立 NDTR-Tree 的上层 R 树;
2. 将各道路对应的下层 R 树均置为空树;
3. **While** (MOD is running) **Do**
4. 接收位置更新消息 LUM (设发送消息的移动对象标识为 *mid*);
5. **If** (LUM仅包含 1 个运行矢量$mv_a=(t_a,(rid_a,pos_a),\vec{v}_a)$) **Then** //LUM是DTTLU或STTLU产生的
6. $RTree_{low}$:=rid_a对应的下层R树;
7. mv_n:=$RTree_{low}$中移动对象*mid*的活动运行矢量;
8. **If** (mv_n=*NULL*) **Then**
9. *Insert*($RTree_{low}$,(*mid*,$\mu(mv_n)$,*mbr*($\mu(mv_n)$)));
10. **Else**
11. *Delete*($RTree_{low}$,(*mid*,$\mu(mv_n)$,*mbr*($\mu(mv_n)$)));
12. *Insert*($RTree_{low}$,(*mid*,$\mu(mv_n,mv_a)$,*mbr*($\mu(mv_n,mv_a)$)));
13. *Insert*($RTree_{low}$,(*mid*,$\mu(mv_a)$,*mbr*($\mu(mv_a)$)));
14. **Endif**;
15. **Else If** (LUM 包含 3 个运行矢量 $mv_{a_1}, mv_{a_2}, mv_{a_3}$) **Then** //LUM 是 IDTLU 产生的

//设 $mv_{a_i}=(t_{a_i},(rid_{a_i},pos_{a_i}),\vec{v}_{a_i})$ ($i\in\{1,2,3\}$)

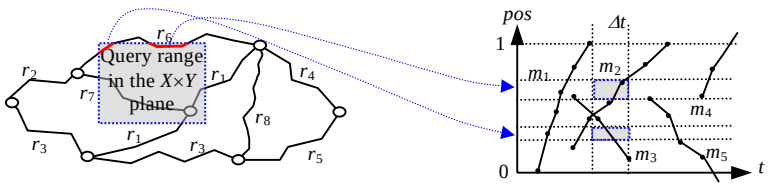
16. $RTree_{low_1}$:= rid_{a_1} 对应的下层 R 树;
17. mv_n:= $RTree_{low_1}$ 中移动对象*mid*的活动运行矢量;
18. *Delete*($RTree_{low_1}$,(*mid*,$\mu(mv_n)$,*mbr*($\mu(mv_n)$)));
19. *Insert*($RTree_{low_1}$,(*mid*,$\mu(mv_n, mv_{a_1})$,*mbr*($\mu(mv_n, mv_{a_1})$));
20. $RTree_{low_2}$:= rid_{a_2} 对应的下层 R 树;
21. *Insert*($RTree_{low_2}$,(*mid*,$\mu(mv_{a_2},mv_{a_3})$,*mbr*($\mu(mv_{a_2},mv_{a_3})$));
22. *Insert*($RTree_{low_2}$,(*mid*,$\mu(mv_{a_3})$,*mbr*($\mu(mv_{a_3})$));
23. **Endif**;
24. **Endif**;
25. **Endwhile**.

NDTR-Tree 中由于包含了活动轨迹单元,因此不仅可以支持对移动对象过去位置的查询,而且可以支持对其当前及将来位置的查询.我们以最常用的区域查询(range query)为例,分析 NDTR-Tree 的查询处理过程.

对于区域查询Q,设其查询输入为$X\times Y\times T$空间中的一个立方体区域$\Delta x\times\Delta y\times\Delta t$.在进行查询处理时,系统将首先根据$\Delta x\times\Delta y$查询NDTR-Tree的上层R树,根据相应原子路段对应的PT_{route}指针找到具体的道路信息,经过计算得到一组(*rid*×*period*)(*period*∈[0,1]且可有多个元素)偶对;然后对每一个(*rid*×*period*)偶对,在对应的下层R树中

© 中国科学院软件研究所 http://www.c-s-a.org.cn

查询与$period\times\Delta t$相交的轨迹单元,并输出相应的移动对象标识.图 7 给出了NDTR-Tree的查询过程.



(a) Querying upper R-Tree and getting ($rid\times period$) pairs (b) Querying lower R-Tree and outputting (m_2,m_3)

(a) 查询上层R树得到($rid\times period$)偶对 (b) 查询下层R树得到输出结果(m_2,m_3)

Fig.7 Range query processing with the NDTR-Tree

图 7 NDTR-Tree 的区域查询处理过程

算法 2 是完整的 NDTR-Tree 查询算法.

算法 2. NDTR-Tree 的区域查询算法.

输入:查询区域$\Delta x\times\Delta y\times\Delta t$;

输出:*Result*; //*mid* 的集合.

1. *Result*←∅;
2. 根据$\Delta x\times\Delta y$ 查询上层 R 树,得到一组偶对 $P=(rid_i, period_i)_{i=1}^{n}$;
3. **For** (*i*:=1 to *n*) **Do**
4. $RTree_{low}$:=rid_i对应的下层R树;
5. 根据$period_i\times\Delta t$查询$RTree_{low}$中的轨迹单元μ;
6. **If** ($\mu\cap(period_i\times\Delta t)\neq\varnothing$) **Then**
7. *Result*:={轨迹单元μ对应的 *mid*}$\cup$*Result*;
8. **Endif**;
9. **Endfor**;
10. **Return** Result.

3 实验比较与分析

为了比较相关算法的性能,我们设计并实现了网络受限移动对象数据库 Net-MODPCS.Net-MODPCS 的服务器部分和移动端部分分别用 C++实现,运行在 Linux 环境下;查询端部分用 Java 实现,可以运行于 Windows 和 Linux 等多种平台环境.在 Net-MODPCS 的基础上,我们实现了本文所介绍的移动对象索引模块,并进行了一系列的实验.

将移动对象索引建立在道路网络基础上的好处在于,可以将交通网络(上层R-Tree)和移动对象动态位置(下层R-Tree)在表示上进行分离,从而使得移动对象在某条道路中的二维移动($X\times Y$)变成一维移动(POS),简化了索引机制并提高了效率.根据文献[12]中的实验结果证明,与直接基于Euclidean空间R树的移动对象索引族相比,基于交通网络的移动对象索引方法具有更好的性能.鉴于以上考虑,我们在实验中选择基于网络的移动对象索引方法作为NDTR-Tree性能比较的参照对象.由于MON-Tree[14]是目前基于网络的移动对象索引方法中性能最好的方法,因此我们选择其作为实验的比较对象.MON-Tree包括以道路为基本单位的组织方式(即route-based MON-Tree,在图 8 中记为Route MON)和以边(边的粒度与原子路段相当,但不带有方向)为基本单位的组织方式(即edge-based MON-Tree,在图 8 中记为Edge MON),本文与这两种组织方式均进行了比较.

通过分析可知,在选取实验数据集时,如下两个因素将会对 MON-Tree 和 NDTR-Tree 的性能产生差异性的影响:(1) 数据集合中平均每条道路所包含的边数ε(影响在表示移动对象跨越不同下层 R 树时的工作量,ε越大则粒度较大的下层 R-Tree 方案的优势越明显);(2) 道路在整个地图空间中的跨度ω(影响 MBR 的重合度,ω越大则粒度较小的上层 R-Tree 方案的优势越明显).而在上述两个方面,北京的交通网络具有典型的代表意义.以平

© 中国科学院软件研究所 http://www.c-s-a.org.cn

均每条道路的边数ε为例,北京数据集约为 2.64,这和大部分数据集是相当的(如文献[14]中为 4.29).而在道路占整个地图空间中的跨度ω方面,北京数据也具有代表意义.基于上述考虑,我们在实验中选择了北京市的真实GIS地图数据来构造交通网络(为了实验方便,我们仅选择了三级以上的道路),并通过一个数据转换程序,将GIS数据转换成了本文所定义的格式.

为了随机地生成移动对象的轨迹,我们开发了一个网络受限移动对象的模拟生成器 NMO-Generator,动态地生成多个网络受限移动对象的运行轨迹.表 1 列出了实验中的主要参数.

Table 1 Parameters of the experiments

表 1 模拟实验的主要参数

Parameter	Value	Meaning
N_{mo}	2000~10000	Number of moving objects
N_{routes}	2 994	Number of routes
N_{edge}	7 917	Number of edges
Lon-Range	116.1~116.7	Longitude range of the map
Lat-Range	39.7~40.1	Latitude range of the map
Window-Num	5 400	Number of query windows
Duration	500 (sec)	Duration of the simulation runs

为了客观地比较 NDTR-Tree 和两种 MON-Tree 在性能上的差异,我们设计了一个通用的程序框架,使得 3 种索引结构可以最大限度地共享程序,并均能动态地维护移动对象的活动运行矢量.基于北京市路网,我们为 2000~10000 个移动对象随机生成移动轨迹,这些轨迹均匀分布在整个路网中.

图 8(a)显示了 500 秒模拟过程中索引的维护性能与移动对象数量的关系.由于索引更新是直接针对下层 R-Tree 进行操作,而 NDTR-Tree 和 route-based MON-Tree 的下层都是以 Route 为单位组织的,因此他们具有相同的维护开销.从图中可以看出,采用 NDTR(或 route-based MON-Tree)进行更新,其维护开销大约相当于 edge-based MON-Tree 的 25%.

我们还统计了平均查询响应时间随移动对象数目增长的关系.将路网的地理区域(经度 116.1~116.7,纬度 39.7~40.1)划分为 90×60=5400 个小区域,在每个区域内用一个 0.014×0.014 的空间窗口进行查询,该窗口尺寸大约可以覆盖 2~3 个边,是典型的区域查询窗口的尺寸.查询的时间范围统一设置为 100~220.通过完成这 5 400 个查询,统计出平均查询响应时间.从图 8(b)可以看出,在不同的移动对象规模下,NDTR-Tree 始终优于 MON-Tree.

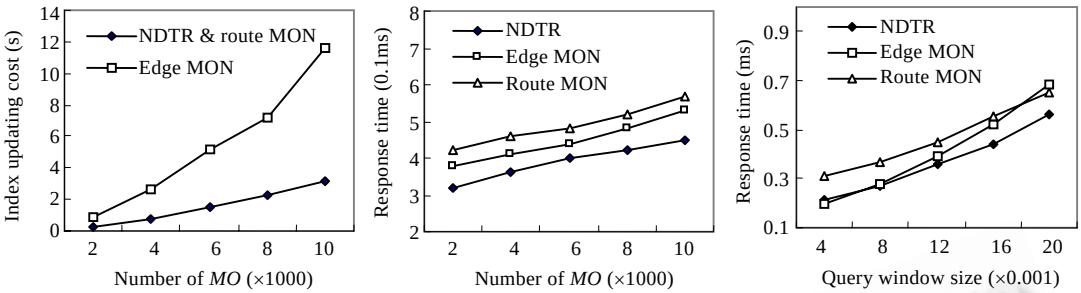
最后,为了分析窗口尺寸对查询性能的影响,我们固定移动对象数目为 6 000,查询时间范围 100~120,然后测试不同窗口尺寸下的平均查询响应时间.选取 0.004~0.02 作为测试窗口尺寸范围.其中,最小窗口覆盖大约一个边,最大窗口覆盖 5~6 个边.实验结果(如图 8(c)所示)表明,当窗口较小时(边长 0.004~0.01),NDTR-Tree 的平均查询性能与 edge-based MON-Tree 基本相当;随着窗口尺寸逐渐增大,NDTR-Tree 响应时间的增长则远远慢于 edge-based MON-Tree;而在各种局部查询窗口尺寸下,NDTR-Tree 始终优于 route-based MON-Tree.

通过以上实验我们可以看出,与 MON-Tree 相比,NDTR-Tree 具有更好的性能,这主要是因为 NDTR-Tree 采用了一种 Hybrid 结构.一方面,上层 R 树以较小粒度的原子路段为索引记录单位,降低了 MBR 的重合,因此可以大大降低回溯所造成的性能开销;另一方面,每棵下层 R 树与粒度较大的道路相对应,从而降低了索引维护及位置更新的开销.

文献[14]虽然指出增加 FNR-Tree 中的粒度(即由 segment-based 过渡到 edge-based 和 route-based)可以减少索引记录的总数,但其所提出的方法是单一粒度的(即上下层 R-Tree 采用相同的粒度),没有考虑本文所提出的混合粒度方案,因此不能有效地兼顾上层 R-Tree 需要较小的粒度以降低 MBR 重合、下层 R-Tree 需要较大的粒度以降低动态维护代价的矛盾.

NDTR-Tree 的另外一个特性是在轨迹单元中存放了速度及方向信息.但事实上,速度和方向仅在计算活动轨迹单元时起作用,当活动运行矢量中所包含的速度方向信息被显式地表示成了到达道路终点的活动轨迹单元片段(我们称此过程为活动运行矢量的显式化)之后,速度方向信息其实就可以删除了,存放速度方向信息只是为了保持运行矢量数据的完整性(如便于从索引中直接提取运行矢量)所采取的可选策略.

© 中国科学院软件研究所 http://www.c-s-a.org.cn



(a) Index updating cost vs. N_{mo} (b) Query response time vs. N_{mo} (c) Query response time vs. query window size

(a) 更新耗时 vs.移动对象数目 (b) 响应时间 vs.移动对象数目 (c) 响应时间 vs.查询窗口尺寸

Fig.8 Performance comparison of NDTR-Tree and MON-Tree

图 8 NDTR-Tree 与 MON-Tree 的性能比较

4 结 论

基于交通网络的移动对象索引是支持海量移动对象管理的一项关键技术.然而,目前的网络受限移动对象索引方法如 FNR-Tree,MON-Tree 等主要针对历史轨迹进行索引,不能支持对运行中的移动对象当前及将来位置的查询,从而极大地影响了其实用性.

为了解决上述问题,本文提出了一种适合于网络受限移动对象数据库的动态轨迹 R 树(network-constrained moving objects dynamic trajectory R-Tree,简称 NDTR-Tree)索引.实验结果表明,NDTR-Tree 提供了良好的查询及维护处理性能.本文的创新点如下:

(1) 提出了一种基于路网的、能够支持移动对象过去、现在以及将来位置的移动对象索引方法.尽管在移动对象的过去、现在以及将来位置的索引方面人们已经做了一些工作(如文献[6,17]),但是这些工作均是基于 Euclidean 空间的.本文的方法不仅探讨了基于路网的完整轨迹索引问题,而且探讨了与位置更新对接的问题;通过对移动对象活动轨迹单元的处理,提供对移动对象过去、现在及将来位置的查询.

(2) 提出了一种基于 Hybrid 结构的移动对象索引 NDTR-Tree.根据详细的文献分析,目前的网络受限移动对象索引方法均是基于单一粒度的.而 NDTR-Tree 通过在上下两层 R-Tree 采用不同的粒度,不仅降低了上层 R 树中 MBR 的重合度,而且还降低了移动对象在跨越不同的下层 R 树时的工作量,从而提高了索引维护及查询处理的效率.

(3) 作为网络受限移动对象数据库研究的一个部分,NDTR-Tree能够与网络受限移动对象模型[18]的位置表示及位置更新方法完全兼容,因此可以无缝地整合到上述数据库模型框架中.

References:

[1] Wolfson O, Xu B, Chamberlain S, Jiang L. Moving objects databases: Issues and solutions. In: Maurizio R, Matthias J, eds. Proc. of the 10th Int'l Conf. on Scientific and Statistical Database Management (SSDBM'98). Capri: IEEE Computer Society, 1998. 111−122.

[2] Saltenis S, Jensen CS, Leutenegger ST, Lopez MA. Indexing the positions of continuously moving objects. In: Chen W, Naughton JF, Bernstein PA, eds. Proc. of the Int'l Conf. on Management of Data. Dallas: ACM Press, 2000. 331−342.

[3] Tao Y, Papadias D, Sun J. The TPR*-Tree: An optimized spatio-temporal access method for predictive queries. In: Johann CF, Peter CL, Serge A, Michael JC, Patricia GS, Andreas H, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 790−801.

[4] Lin B, Su J. On bulk loading TPR-Tree. In: Proc. of the 5th IEEE Int'l Conf. on Mobile Data Management (MDM 2004). Berkeley: IEEE Computer Society, 2004. 114−124.

[5] Tung HDT, Jung YT, Lee EJ, Ryu KH. Moving point indexing for future location query. In: Shan W, Dongqing Y, Katsumi T, Fabio G, Shuigeng Z, Eleni EM, Tok WL, Il-Yeol, Jihong G, Heinrich CM, eds. Proc. of the Conceptual Modeling for Advanced

© 中国科学院软件研究所 http://www.c-s-a.org.cn

Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO. Shanghai: Springer-Verlag, 2004. 79−90.

[6] Liu ZH, Liu XL, Ge JW, Bae HY. Indexing large moving objects from past to future with PCFI±Index. In: Jayant RH, Vijayaraman TM, eds. Proc. of the 11th Int'l Conf. on Management of Data. Goa: Computer Society of India, 2005. 131−137.

[7] Kim D, Shin J, Kang H, Han K. An efficient location index for the semantic search of moving objects. In: Roman O, Yunmook Nah, Peter PP, Franz-Josef R, eds. Proc. of the Software Technologies for Embedded and Ubiquitous Systems (SEUS 2007). Santorini: Springer-Verlag, 2007. 516−526.

[8] Pfoser D, Jensen CS, Theodoridis Y. Novel approach to the indexing of moving object trajectories. In: Abbadi AE, Brodie ML, Chakravarthy S, Dayal U, Kamel N, Schlageter G, Whang KY, eds. Proc. of the 26th Int'l Conf. on Very Large Data Bases. Cairo: Morgan Kaufmann Publishers, 2000. 395−406.

[9] Pfoser D. Indexing the trajectories of moving objects. IEEE Data Engineering Bulleetin, 2002,25(2):3−9.

[10] Tao Y, Papadias D. The MV3R-tree: A spatio-temporal access method for timestamp and interval queries. In: Apers P, Atzeni S, Paraboschi S, Ramanohamnarao K, Snodgrass RT, eds. Proc. of the 27th Int'l Conf. on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001. 431−440.

[11] Ding R, Meng XF, Bai Y. Efficient index maintenance for moving objects with future trajectories. In: Proc. of the 8th Int'l Conf. on Database Systems for Advanced Applications. Kyoto: IEEE Computer Society, 2003. 183−194.

[12] Frentzos E. Indexing objects moving on fixed networks. In: Hadzilacos T, Manolopouls Y, Roddick JF, Theodoridis Y, eds. Proc. of the 8th Int'l Symp. on Spatial and Temporal Databases (SSTD). Santorini: Springer-Verlag, 2003. 89−305.

[13] Pfoser D, Jensen CS. Indexing of network constrained moving objects. In: Proc. of the 11th ACM Int'l Symp. on Advances in Geographic Information Systems. New Orleans: ACM, 2003. 25−32.

[14] Almeida VT, Güting RH. Indexing the trajectories of moving objects in networks. GeoInformatica, 2005,9(1):33−60.

[15] Chen J, Meng XF. Indexing future trajectories of moving objects in a constrained network. Journal of Computer Science and Technology, 2007,22(2):245−251.

[16] Guo J, Guo W, Zhou DR. Indexing approach for querying about present and future based on constrained moving objects in spatial-temporal databases. Journal of Chinese Computer Systems, 2007,28(2):128−131 (in Chinese with English abstract).

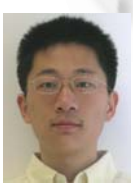
[17] Pelanis M, Saltenis S, Jensen CS. Indexing the past, present and anticipated future positions of moving objects. ACM Trans. on Database Systems, 2006,31(1):255−298.

[18] Ding Z, Güting RH. Managing moving objects on dynamic transportation networks. In: Proc. of the 16th Int'l Conf. on Science and Statistical Database Management (SSDBM 2004). Santorini: IEEE Computer Society, 2004. 287−296.

附中文参考文献:

[16] 郭菁,郭薇,周洞汝.基于受限移动对象当前及将来时刻检索的时空索引结构研究.小型微型计算机系统,2007,28(2):128−131.

丁治明(1966－),男,湖北江陵人,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为数据库与知识库系统,移动计算,信息检索.



李肖南(1983－),男,硕士生,主要研究领域为移动数据库.

余波(1978－),男,博士,主要研究领域为数据库与分布式系统.

© 中国科学院软件研究所 http://www.c-s-a.org.cn